PATENT ABSTRACTS OF JAPAN

(11)Publication number : 04-362738
(43)Date of publication of application : 15.12.1992

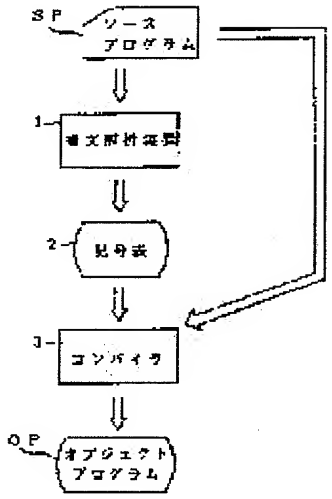
(51)Int.Cl. G06F 9/45

(21)Application number : 03-165080
(22)Date of filing : 10.06.1991

(71)Applicant : OKI ELECTRIC IND CO LTD
(72)Inventor : HARADA MINORU
NISHIKAZE HAJIME

(54) VARIABLE CONTROL METHOD

(57)Abstract:
PURPOSE: To reduce the time and the labor required for the coding operation and to decrease the debugging tasks by describing a syntax to point an area where a variable storing the integer data is coded.
CONSTITUTION: A source program SP which is generated on the assumption of, the working with a specific computer and defines a store form proper to a specific computer for the integer data to a variable is shifted to another computer having a different store form of the integer date. Under such conditions, the area pointing symbols SW and EW are described on the program SP to specify the area where the variable is described. Then the store form of the integer data is decided for all variables described on the program SP so that the program SP is converted into an object program OP. The object codes are added so as to change the store forms for the variable described in the areas pointed by both symbols SW and EW. Thus a program OP is generated. As a result, the variable store form is automatically changed by a compiler 3.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平4－362738

(43)公開日　平成４年(1992)12月15日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/45				
		9193－5Ｂ	Ｇ０６Ｆ　9／44　３２２　Ａ	

審査請求　未請求　請求項の数１(全　５　頁)

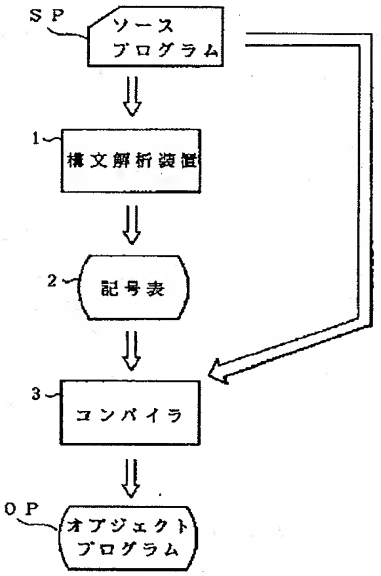
(21)出願番号　特願平3－165080

(22)出願日　平成３年(1991)６月10日

(71)出願人　000000295
沖電気工業株式会社
東京都港区虎ノ門１丁目７番12号
(72)発明者　原田　稔
東京都港区虎ノ門１丁目７番12号　沖電気工業株式会社内
(72)発明者　西風　一
東京都港区虎ノ門１丁目７番12号　沖電気工業株式会社内
(74)代理人　弁理士　佐藤　幸男

(54)【発明の名称】　変数管理方法

(57)【要約】
【目的】　変数の格納形式が相違してもソースプログラムＳＰの移植を手間をかけることなく容易に行なうことができる変数管理方法を提供する。
【構成】　予め、ソースプログラムＳＰ上に、整数データを格納する変数の記述された領域を特定する領域指示記号ＳＷ，ＥＷを記述する。ソースプログラムＳＰをオブジェクトプログラムＯＰに変換する際、領域指示記号ＳＷ，ＥＷで指示された領域に記述された変数について、格納形式を変更するためのオブジェクトコードを付加する。



本発明に係る処理説明図

1

【特許請求の範囲】

【請求項1】　特定の計算機上での稼働を想定して生成され、変数への整数データの格納形式が前記特定の計算機に固有の形態を採るソースプログラムを、前記変数への前記整数データの格納形式が異なる他の計算機に移植する場合、前記ソースプログラム上に、前記変数の記述された領域を特定する領域指示記号を記述し、前記ソースプログラムをオブジェクトプログラムに変換するために、当該ソースプログラム上の全ての変数について整数データの格納形式を判断し、前記領域指示記号で指示された領域に記述された変数について、前記格納形式を変更するためのオブジェクトコードを付加し、前記オブジェクトプログラムを生成することを特徴とする変数管理方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、変数への整数データの格納形式の異なる計算機の相互で、同一のソースプログラムの利用を実現する変数管理方法に関する。

【０００２】

【従来の技術】計算機を動作させるためには、プログラムが必要となる。このプログラムは、計算機の機種や使用言語によって、種々の形式のものを用意することができる。またプログラムには、必ずといって良いほど任意のデータを格納する変数が用意される。変数には、文字データや整数データ等、各種のデータを格納することができる。そして整数データを格納する変数の場合、ビッグエンディアンとリトルエンディアンと呼ばれる２種類の格納形式が用意されている。

【０００３】図２に、一般的な変数の説明図を示す。ここでは、整数データとして、１６進数“12345678”の整数データを格納する変数Ｈが８バイト、そして、変数Ｈの読取りが２バイト単位で実行されるものとする。この場合、変数Ｈがリトルエンディアンの格納形式を採る場合、変数Ｈには、整数データが“12345678”の順番で、即ち若番地側（ＬＳＢ側）に整数値の下位桁が格納される。

【０００４】一方、変数Ｈがビッグエンディアンの格納形式を採る場合、変数Ｈには、整数データが“78563412”の順番で、即ち若番地側（ＬＳＢ側）に整数値の上位桁が格納される。これらの格納形式は計算機（オペレーティングシステム）固有のもので、ソースプログラムを開発する際、考慮しなければならない事柄の一つである。例えばＣ言語において変数には、その定義を行なうプログラム上の位置により、グローバル変数とローカル変数（自動変数）の２つを適宜用意することができる。グローバル変数の場合、存在区間がプログラムの全体に渡り、ローカル変数の場合、存在区間は限定された区間（例えば特定の関数）内に制限される。ローカル変数の内容を、定義がなされた関数以外の他の関数に渡す場合、一般にローカル変数の内容が格納されたメモリ上の位置を示すアドレス演算を行ない、この結果をポインタ変数として通知することにより、ローカル変数の内容の受渡しを実現することができる。

2

【０００５】さて、従来ポインタ変数の通知を受けて、ローカル変数の値を受継ぐ場合、格納形式が問題となっていた。即ち、リトルエンディアンに対応した計算機上で、ビッグエンディアンに対応したプログラムを実行させ、ポインタ変数の示す内容の読取りを行なう場合、整数データの上位桁と下位桁の順序が乱れ、全く意味の無い整数データとして取扱う恐れがあった。このため、ソースプログラムをリトルエンディアン及びビッグエンディアンの計算機の両者で利用しようとした場合、変数をリトルエンディアンで取扱うか、ビッグエンディアンで取扱うかの切り分けを行ない、それぞれ変数の読取り形態を変更する必要がある。

【０００６】図３に、従来一般の切り分けに係る説明図を示す。図はＣ言語で記述されたソースプログラムの例で、ここでは、変数lengthが４バイトの容量を持ち、リトルエンディアンの場合、１バイト単位でＬＳＢ側から変数の認識を行ない、ビッグエンディアンの場合、１バイト単位でＭＳＢ側から変数の認識を行なうものとする。まず、ステップＳ１において、計算機がビッグエンディアンであるかの判定を行ない、判定が肯定された場合、ステップＳ２において、変数lengthのＭＳＢ側から１バイトずつ読み出して、順次ｃｂｕｆ［0～3］に格納し、ステップＳ５において処理を終了する。

【０００７】ステップＳ１において、判定が否定された場合、処理がステップＳ３に移り、ステップＳ４において、変数lengthのＭＳＢ側から１バイトずつ読み出し、順次ｃｂｕｆ［3～0］に格納し、ステップＳ５において処理を終了する。

【０００８】

【発明が解決しようとする課題】変数の格納形式の異なる計算機相互にソースプログラムを移植しようとした場合、外部関数に変数の値を受渡す処理が実行されるソースプログラム上に、図３において説明したような変換用のソースプログラムのコーディングが必ず必要となる。このため、コーディングの手間の増加及びソースプログラムの読解性が損なわれてしまうといった問題が生じていた。また、変換用のソースプログラムのコーディングを忘れた場合、デバッグ作業を強いられるといった問題も生じていた。本発明は以上の点に着目してなされたもので、変数の格納形式が相違してもソースプログラムの移植を手間をかけることなく容易に行なうことができる変数管理方法を提供することを目的とする。

【０００９】

【課題を解決するための手段】本発明の変数管理方法は、特定の計算機上での稼働を想定して生成され、変数への整数データの格納形式が前記特定の計算機に固有の

3

形態を採るソースプログラムを、前記変数への前記整数データの格納形式が異なる他の計算機に移植する場合、前記ソースプログラム上に、前記変数の記述された領域を特定する領域指示記号を記述し、前記ソースプログラムをオブジェクトプログラムに変換するために、当該ソースプログラム上の全ての変数について整数データの格納形式を判断し、前記領域指示記号で指示された領域に記述された変数について、前記格納形式を変更するためのオブジェクトコードを付加し、前記オブジェクトプログラムを生成する。

【００１０】

【作用】予め、ソースプログラム上に、整数データを格納する変数の記述された領域を特定する領域指示記号を記述する。ソースプログラムをオブジェクトプログラムに変換する際、領域指示記号で指示された領域に記述された変数について、格納形式を変更するためのオブジェクトコードを付加する。

【００１１】

【実施例】図１に、本発明に係る処理説明図を示す。なお、ここではビッグエンディアンの計算機を想定したソースプログラムを、リトルエンディアンの計算機に移植する場合を例に説明する。図に示すように、ソースプログラムＳＰは、構文解析装置１に内容が解析され、ソースプログラムＳＰ上の整数を格納する変数について、格納形式がビッグエンディアンであるかリトルエンディアンであるかが判定され、判定結果を格納する記号表２が生成される。コンパイラ３は、記号表２を参照し、ソースプログラムＳＰのコンパイルを実施して、計算機が認識して処理を実行するために参照するオブジェクトプログラムＯＰの生成を行なう。

【００１２】ソースプログラムＳＰには、予めビッグエンディアンを想定して記述された箇所を特定するための構文（領域指示記号）をコーディングする。図４に、本発明に係る構文のコーディング例を示す。図に示すように、ステップＳ１２の範囲にコーディングされた内容が、ビッグエンディアンを想定したもので、その前後、即ちステップＳ１１及びステップＳ１３において、ビッグエンディアンを想定してコーディングされた箇所の始まりと終わりを示す開始構文ＳＷ及び終了構文ＥＷをコーディングする。構文解析装置１は、ソースプログラムＳＰ上で宣言された変数の全てについて、記号表にデータの格納形態と共に登録する。そして、開始構文ＳＷ及び終了構文ＥＷにより指示された箇所で宣言され、整数を格納する変数については、構文の指示する格納形態、この場合、ビッグエンディアンを想定してコーディングされた変数であることを示す登録を記号表２に行なう。

【００１３】図５に、記号表２の説明図を示す。先に図４において説明したステップＳ１２において、整数データを格納する変数として、変数lengthが宣言されている。このため、構文解析装置１は、変数lengthに対応して

4

て変数の格納形式、即ち属性としてビッグ（ビッグエンディアン）の登録を行なう。なお、開始構文ＳＷ及び終了構文ＥＷにより指示された箇所以外の変数（例えば変数ｉ）については、計算機の機能に合わせて、属性としてリトル（リトルエンディアン）の登録が実施される。

【００１４】コンパイラ３は、構文解析装置１による記号表２の生成が完了すると、ソースプログラムＳＰのコンパイルを実施することになる。この際、記号表２を参照して、属性がビッグに設定された記号（変数length）を認識すると、変数lengthに係る処理部分に、ビッグエンディアンからリトルエンディアンに格納形式を変換するための記述、即ち、先に図３において説明したステップＳ４に対応する内容のオブジェクトコード（機械語）を自動的に付加し、オブジェクトプログラムＯＰを生成する。このオブジェクトコードは、先に図３のステップＳ４に示したコーディングに対応するものである。

【００１５】以上の説明のように、予めソースプログラムＳＰ上に所定の構文をコーディングし、この構文に指示された箇所の整数データを格納する変数を特定することにより、コンパイラ３による変数の格納形式の変換を自動的に行なうことができる。本発明は、以上の実施例に限定されない。実施例では、Ｃ言語のソースプログラムを例に説明を行なったが、プログラムの言語は特に限定されるものでなく、例えばアセンブラ言語等おいても本発明を適用することができる。また、ビッグエンディアンを想定したソースプログラムを、リトルエンディアンの計算機に移植する場合を例に説明したが、リトルエンディアンを想定したソースプログラムを、ビッグエンディアンの計算機に移植する場合にも本発明を適用することができる。

【００１６】

【発明の効果】以上説明したように本発明によると、整数データを格納する変数がコーディングされた箇所を指示する構文を記述することにより、その後自動的に格納形式の変換が実施されるため、整数データを格納する変数の個々について、格納形式を変換するためのコーディングを実施する必要がない。このため、コーディングに要する手間の軽減及びコーディングを忘れたためのデバッグ作業を減少させることができる。

【図面の簡単な説明】

【図１】本発明に係る処理説明図である。

【図２】一般的な変数の説明図である。

【図３】従来一般の切り分けに係る説明図である。

【図４】本発明に係る構文のコーディング例である。

【図５】本発明に係る記号表の説明図である。

【符号の説明】

１　構文解析装置

２　記号表

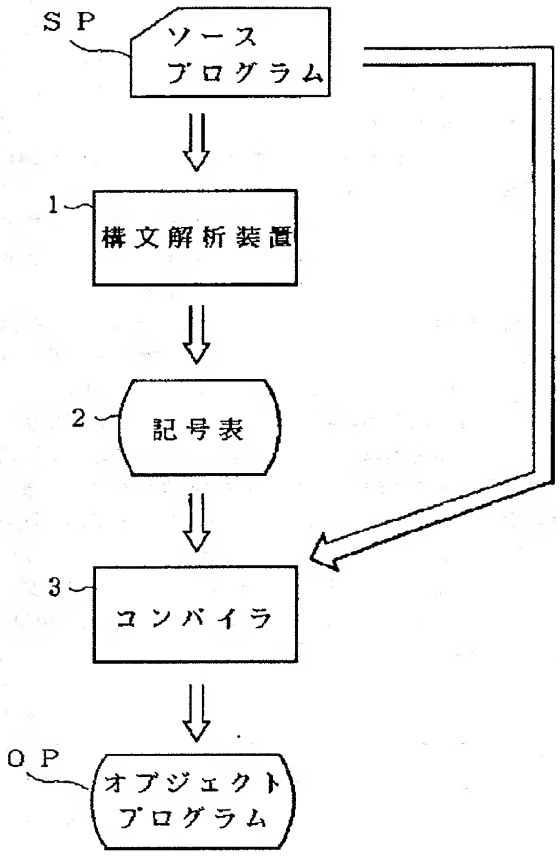
３　コンパイラ

ＳＰ　ソースプログラム

5　　6

OP　オブジェクトプログラム

【図1】



本発明に係る処理説明図

【図5】

記号	属性
i	リトル
length	ビッグ
⋮	⋮

記号表の説明図

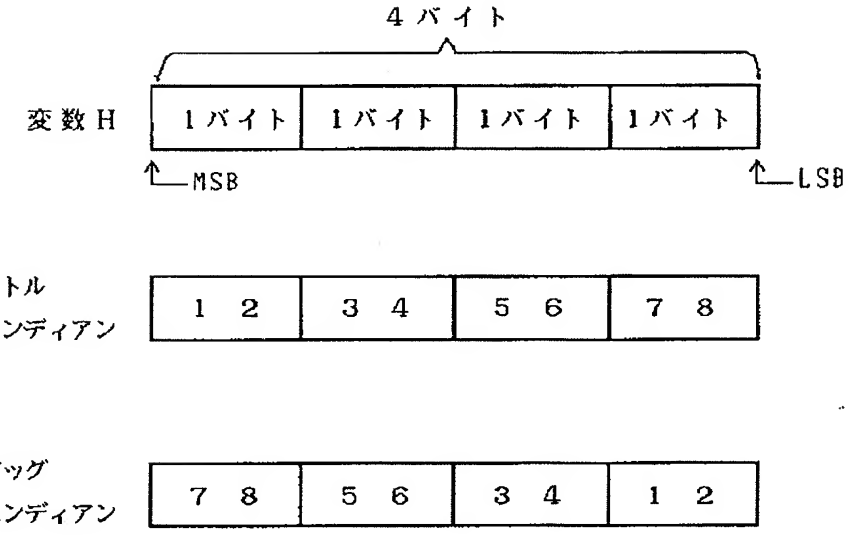
【図3】

```
S1～#ifdef Big
S2～   length = cbuf[0]<<24+cbuf[1]<<16+cbuf[2]<<8+cbuf[3];
S3～#else
S4～   length = cbuf[3]<<24+cbuf[2]<<16+cbuf[1]<<8+cbuf[0];
S5～#end
```

従来一般の切り分けに係る説明図

【図2】

整数データ　　（１２３４５６７８）HEX



一般的な変数の説明図

【図4】

```
                                          SW
S11～ #pragma    B I G    E N D I A N
      typedef    struct    dl_data {
                              int       length;
S12                           char      data[256];
      } D1_D A T A ;

S13～ #pragma    E N D    B I G    E N D I A N
                                                 EW
```

本発明に係る構文のコーディング例